# tokenizer-4

November 14, 2024

[1]:
```
import os
from tqdm import tqdm
import pandas as pd
import argparse
from tokenizers import SentencePieceBPETokenizer
from transformers import PreTrainedTokenizerFast
import argparse
import datetime
import pandas as pd
```

[2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[3]:
```
def train_tokenizer(data_list, vocab_size=32768, model_name="test"):

    ## Change bos & eos
    bos_tok = "<bos>"
    eos_tok = "<eos>"

    ## Add basic characters to this below list, including numbers & special␣
 ↪language characters.
    special_char = ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9"]


    tokenizer = SentencePieceBPETokenizer()

    tokenizer.train_from_iterator(
        data_list,
        vocab_size = 50000,
        min_frequency = 5,
        special_tokens = ["<pad>", "<unk>", bos_tok, eos_tok, "<user>",␣
 ↪"<assistant>"] + special_char,
        show_progress = True,
    )
```

```
    ## Don't forget to add special tokens.
    transformer_tokenizer = PreTrainedTokenizerFast(
        tokenizer_object=tokenizer,
        bos_token = bos_tok,
        eos_token = eos_tok,
        unk_token = "<unk>",
        pad_token = "<pad>",
        mask_token = "<mask>",
        padding_side = "left",
        truncation_side = "right",
        additional_special_tokens = ["<user>", "<assistant>"],
        clean_up_tokenization_spaces = False,
    )

    transformer_tokenizer.save_pretrained(model_name)
```

[4]:
```
### Importing Data
df = pd.read_csv("/content/drive/MyDrive/sample_set4.csv")
# df_2 = pd.read_csv("English_2.csv")
```

[5]:
```
cleaned_data = []
for i in range(len(df["content"])):
  value = str(df['content'][i]).replace('\n','') # Convert to string before␣
 ↪replacing
  cleaned_data.append(value)
df['clean_content'] = cleaned_data
```

[6]:
```
df.head()
```

[6]:

| | filename | content | clean_content |
|---|---|---|---|
| 0 | full_text_4206.txt | * \n\n\nV \n\n\n. \n\n . \n\n\n … | * V . . … |
| 1 | full_text_4207.txt | \n\n\n 8 8 ^ ' … | 8 8 ^ ' … |
| 2 | full_text_4208.txt | \n\n\n - . \n\n… | - . \|. … |
| 3 | full_text_4209.txt | / \n\n\n;• \n\n\n \n\n\n - … | / ;• - 1- … |
| 4 | full_text_4215.txt | , \n\n\n 1№ I \n\n\n 1 … | , 1№ I 1 , … |

[7]:
```
len(df["clean_content"].to_list())
```

[7]:
```
545
```

[8]:
```
df["clean_content"] = df["content"].astype(str)
```

[9]:
```
from sklearn.model_selection import train_test_split
```

[10]:
```
train_texts, test_texts = train_test_split(df["clean_content"].to_list(),␣
↪test_size=0.2, random_state=42)
```

[11]:
```
# Train the tokenizer on the training data
tokenizer = train_tokenizer(train_texts, vocab_size=32000,␣
↪model_name="test_tokenizer")
```

[12]:
```
### Testing Training Tokenizer
from transformers import AutoTokenizer
```

[13]:
```
tokenizer = AutoTokenizer.from_pretrained("test_tokenizer")
```

[14]:
```
len(tokenizer.get_vocab())
```

[14]:
```
50001
```

[15]:
```
# Tokenize the input text using tokenizer()
tokens = tokenizer(test_texts, add_special_tokens=True,␣
↪return_tensors=None)['input_ids']

# Calculate the total number of tokens
num_tokens = sum(len(token_list) for token_list in tokens)

# Calculate the total number of words in the Series
num_words = sum(len(text.split()) for text in test_texts)

# Calculate the fertility score
fertility_score = num_tokens / num_words

print(f"Total number of tokens: {num_tokens}")
print(f"Total number of words: {num_words}")
print(f"Fertility score: {fertility_score:.2f}")
```

```
Total number of tokens: 25011532
Total number of words: 9690240
Fertility score: 2.58
```

[ ]:
```

```